(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 January 2001 (18.01.2001)

PCT

(10) International Publication Number
WO 01/04770 A2

(51) International Patent Classification[7]: G06F 15/80

(21) International Application Number: PCT/US00/18939

(22) International Filing Date: 11 July 2000 (11.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/143,445 13 July 1999 (13.07.1999) US

(71) Applicant *(for all designated States except US)*: **ALTEON WEB SYSTEMS, INC.** [US/US]; 50 Great Oaks Road, San Jose, CA 95119 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **LEE, Keith** [US/US]; 1447 Sajak Avenue, San Jose, CA 95131 (US). **SCHMALTZ, Dean** [US/US]; 541 Tramway Drive, Milpitas, CA 95035 (US).

(74) Agents: **GLENN, Michael** et al.; Glenn Patent Group, 3475 Edison Way, Ste. L, Menlo Park, CA 94025 (US).

(81) Designated States *(national)*: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

**Published:**

— *Without international search report and to be republished upon receipt of that report.*

*For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.*

WO 01/04770 A2

(54) Title: METHOD AND ARCHITECTURE FOR OPTIMIZING DATA THROUGHPUT IN A MULTI-PROCESSOR ENVIRONMENT USING A RAM-BASED SHARED INDEX FIFO LINKED LIST

(57) Abstract: A method and architecture for optimizing data throughput in a multiprocessor environment makes use of a RAM-based, shared index FIFO linked list, in which data to be processed is written to a central buffer and the index FIFO, constituting a linked list of indexes to the buffered data is passed between processing units within the system, providing a substantial reduction in the gate count required for processing the data. Messages are written to a central buffer; a linked list of indexes to the messages is created, and then pipelined to a processing unit as an index FIFO, so that the processor reads the entries of the linked list in sequence; as the entries are read, a message indicated by the entry is processed. Entries are enqueued and dequeued in an index FIFO RAM, so that enqueuing and dequeuing are performed in a single cycle with a single write operation.

# METHOD AND ARCHITECTURE FOR OPTIMIZING DATA THROUGHPUT IN A MULTI-PROCESSOR ENVIRONMENT USING A RAM-BASED SHARED INDEX FIFO LINKED LIST

## BACKGROUND OF THE INVENTION

### FIELD OF THE INVENTION

The present invention relates to data transfer in a computer system. More particularly, the invention relates to a method and architecture for optimizing data throughput in a multi-processor environment by writing data to be processed to a central buffer and passing the various processors a FIFO-like data structure constituting a linked list of indexes to the buffered data.

### DESCRIPTION OF PRIOR ART

In the data processing art, it is an exceedingly common operation to pass data from one processing system to another. The data may be passed from process to process within a single processor, or between processing units in a multiple processor environment. Passing data between processing systems requires that the data be repeatedly copied to each processing system. The art provides various systems and methods for accomplishing data transfer in this manner

For example, P. Chambers, S. Harrow, *Virtual contiguous FIFO for combining multiple data packets into a single contiguous stream*, U.S. Patent No. 6,016,315 (January 18, 2000) describe an arrangement in which data packets are supplied to a DSP from a PCI bus through a plurality of FIFO RAM units operating in parallel. R. Panwar, *System for efficient implementation of multi-ported logic structures in a processor*, U.S. Patent No. 6,055,616 (April 25, 2000) describes a system and method for efficient implementation of a multi-port logic first-in, first-out structure that provides for reduced on-chip area requirements. A common feature of both disclosed systems is that data must be transferred between units by copying, creating a potential bottleneck, and wasting I/O bandwidth and memory bandwidth. Accordingly, it would be advantageous to provide a means for avoiding copying of data between processors.

R. Fishler, B. Zargham, *System for transferring a data stream to a requestor without copying data segments to each one of multiple data source/sinks during data stream building*, U.S. Patent No. 5,941,959 (August 24, 1999) describe a method for getting descriptors to data and passing the descriptors to data

sources and sinks, thereby avoiding copying the data among the data sources and sinks. The data descriptors are organized into a queued I/O data structure comprising a doubly linked list. R. Baumert, A. Seaman, S. Steves, *Method and apparatus for optimizing the transfer of data packets between local area networks*, U.S. Patent No. 6,067,300 (May 23,2000) describe a switch apparatus having a packet memory, a packet descriptor memory that stores pointers to the stored data packets and buffered data paths employing FIFO buffers. The FIFO buffers utilize conventional queued data structures. While the described methods effectively avoid copying of data, both inter- and intra-processor, conventional methods of adding to and removing data descriptors from the queues are employed, requiring the allocation of an entry and a pointer, and a subsequent read-write-modify operation. It would be highly desirable to further reduce processing overhead by streamlining enqueue and dequeue operations.

## SUMMARY OF THE INVENTION

The invention provides a method and architecture for optimizing data throughput in a multiprocessor environment through the use of a RAM-based, shared index FIFO linked list, in which data to be processed is written to a central buffer and the index FIFO, constituting a linked list of indexes to the buffered data is passed between processing units within the system. The invention advantageously reduces the overhead required to process a data stream in a variety of ways. First, since the FIFO is composed of indexes, rather than the actual data, a significant reduction in gate count required for processing is achieved. Second, the use of a FIFO-like structure, rather than a conventional pipeline, greatly reduces pipeline interlock; and third, the use of a FIFO-like linked list, instead of a FIFO, frees the system of the requirement, imposed by a conventional FIFO, of processing data frames in sequence. A novel method of dequeuing and enqueuing linked list entries enables entries to be enqueued and dequeued in a single cycle, with a single read, rather than the conventional read-modify-write method in common use. In general, the invented method involves the steps of: providing messages to be processed; writing the data messages to a central buffer; creating a linked list of indexes to the messages, where an index constitutes a pointer to a buffer address occupied by a specific message, and where an index constitutes an entry in said linked list, with each entry also including an index pointer to a next entry in said linked list; pipelining the linked list to a processing unit as an index FIFO, so that the processor reads the entries of the

linked list in sequence; as the entries are read, processing a message indicated by said entry; and enqueuing and dequeuing the entries in an index FIFO RAM, so that enqueuing and dequeuing are performed in a single cycle with a single write operation.

The invention is also embodied as an architecture, the architecture including one or more processing units, the aforementioned central buffer and a RAM-based, shared index FIFO linked list; one or more pipelines for feeding the linked list to the processing units; and the afore-mentioned index FIFO RAM, wherein the linked lists are stored and entries dequeued and enqueued.

## BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 provides a block diagram of an architecture for optimizing data throughput in a multiprocessor environment, according to the invention;

Figure 2 provides a diagram of a linked list of indexes, according to the invention; and

Figure 3 provides an index FIFO RAM memory map, according to the invention.

## DETAILED DESCRIPTION

The invention provides a method and architecture for optimizing data throughput in a multiprocessor environment through the use of a RAM-based, shared index FIFO linked list, in which data to be processed is written to a central buffer and the index FIFO, constituting a linked list of indexes to the buffered data, is passed between processing units within the system. Several noteworthy advantages are provided by the invention:

- Since the FIFO is composed of indexes, rather than the actual data, a significant reduction in gate count required for processing is achieved;
- The use of a FIFO-like structure rather than a conventional pipeline, greatly reduces pipeline interlock;
- The use of a FIFO-like linked list, instead of a FIFO, frees the system of the requirement, imposed by a conventional FIFO, of processing data frames in sequence;

- A novel method of dequeuing and enqueuing linked list entries enables entries to be enqueued and dequeued in a single cycle, with a single read, rather than the conventional read-modify-write method in common use.

An example is provided to illustrate the dramatic reduction in gate count achievable with the invention: In the preferred embodiment, the invention is implemented in a network switch, for forwarding data frames over an IP network. Incoming data frames are matched with records of previous forwarding results to determine a next hop for each of the data frames. A typical forwarding result record is approximately sixty-four bits; however an index pointer to that forwarding result record is only six bits. Therefore, the pronounced increase in throughput through the use of FIFO-like linked list of indexes, instead of a FIFO of actual data frames or results will be apparent to those skilled in the art. While the invention as described herein is implemented in a network switch, other implementations are possible. The invention finds application in any data processing environment in which a data stream is passed between processes or processing units.

Referring now to Figure 1, shown is an architecture for optimizing data throughput in a multiprocessor environment through the use of a RAM-based, shared index FIFO linked list. A network switch 10 receives incoming data frames at an ingress port (not shown). The first sixty-four bits of each frame constitute the header. The headers are stored in a header buffer RAM 12. In the preferred embodiment, the header buffer RAM is implemented as a 256 x 64 bit dual port RAM, organized as 8 x 64 bits per frame, which allows for a total of 32 frame header buffers. The write port is designed as a thirty-two data frame buffer FIFO, and the read port is designed to be randomly accessed by the various processing units. This description of the header buffer RAM is exemplary only, and is not intended to limit the invention. Other schemes for organizing the buffer RAM will be apparent to those skilled in the art. The remainder of the frame and records of previous forwarding results are stored in a working RAM 11. From the time that a data frame is received at an ingress port to the time that it is routed to a next hop, the data frame is processed in a serial fashion by one or more processing units 14 within the network switch. Typically, processing will include reading a source address and a destination address from the frame header, searching a variety of data structures to find a forwarding result with a destination address matching that of the frame header, and possibly modifying the frame header. In order to pipeline data frames and results from one processing unit to

another, each header buffer has a set of associated bytes reserved in the working RAM 11 that are used to pass information between the various processing units 14 of the switch 10.

Conventionally, when a data stream is pipelined, the actual data, or result, is passed from pipeline stage to pipeline stage. However, if the results and the data are housed in a central location, and an index of pointers instead are passed from pipeline stage to pipeline stage, the required gate count for processing is substantially reduced, as previously illustrated. Thus, the invention provides a linked list to index frame data stored in the header buffer RAM. The implementation of linked lists is well known to those skilled in the art of computer programming and software design. An entry in the linked list of the invention, shown in Figure 2, includes a pointer to a specific header buffer 20 plus an index pointer to the next entry in the linked list 21. The linked list also includes a head pointer 22 to designate the first entry in the list and a tail pointer 23, to designate the final entry of the list. In the preferred embodiment of the invention, each linked list includes thirty-two entries to correspond to the thirty-two locations of the header buffer RAM. In addition, each list also includes an empty entry at the tail as a placeholder, for a total of thirty-three entries per linked list. Other implementations of the linked list consistent with the spirit and scope of the invention will be apparent to those skilled in the art. The function of the placeholder entry will be described in detail further below. Thus, each processing unit may operate from a pipeline of these indexes, significantly reducing the overhead of processing the data stream by reducing gate count.

A further gate reduction is achieved, as compared to a convention FIFO of indexes, by sharing the linked list entries among processing queues. In a multi-processor system that doesn't share entries among processing queues, a conventional FIFO that allows up to a maximum number of frames, $x$, requires $x$ entries for each processing unit. Therefore, an exemplary system having eight processing units, where $x = 32$, would require a total of 256 (32 x 8) entries. Utilizing a shared linked list requires only forty entries: $x$ + the number of processing units, or 32 + 8.

However, the invention includes an additional enhancement. In the interest of maximizing data throughput, it is desirable to free up header buffers as quickly as possible. Due to the interlocking nature of a pipeline, however, data frames may not be processed out of the sequence imposed by the various

stages of the pipeline. Thus, a frame may not proceed to the next stage of the pipeline, until the frame preceding it has cleared that stage. In the present invention, the linked list is passed between processing units in the manner of a FIFO. Processing the linked list as a FIFO allows the processing of the entries of the list to proceed independently of the processing of the corresponding frame. Therefore, if processing of an earlier frame takes longer due to the size of the frame, subsequent frames may still be processed, because processing of the corresponding index in the linked list is allowed to proceed, unhampered by a delay imposed by the processing of the larger frame. Furthermore, processing of the linked list is allowed to proceed, unimpeded by bottlenecks that may be created due to memory latency, for example when a processing unit fetches a data frame from the working RAM for processing. Processing the linked list as a FIFO yields yet another advantage. The characteristics of the FIFO allow all stages of the pipeline to be decoupled from each other, so that a delay in processing of a later frame does not create a bottleneck that prevents preceding frames from moving forward.

In the preferred embodiment of the invention, a linked list is provided for each processing unit. All linked lists are stored in an index FIFO RAM unit 13. Figure 3 provides a map of an exemplary index FIFO RAM. As previously indicated, the shared index FIFO linked list is RAM-based, meaning that all operations to the linked lists occur in RAM. Operations on the linked list include dequeuing and enqueuing. As shown in Figure 2, the serial arrangement of the various processing units creates a data flow in which a head entry from a linked list for a first processing unit 24 is dequeued and enqueued to the tail of a second processing unit 25. In conventional implementations of linked lists, dequeue and enqueue operations may be register-based or RAM based. The empty entry allocated at the tail of each linked list, previously described, allows the current invention to enqueue by writing an entry dequeued from the head of another list to the empty record, and reusing the index pointer as the new tail pointer. Enqueue and dequeue operations are performed in the index FIFO RAM by enqueue and dequeue units (not shown), respectively. Listed below are the steps involved in dequeuing from a first linked list to a second linked list.

- {HeaderBufIdxA, FifoANxtIdxFifo} ⇐ IdxFifo[FifoAHeadPtr]. Load BufIdx and next entry pointed by the head pointer. When done with processing, continue with next step for dequeue operation.

- IdxFifo[FifoBTailPtr] ⇐ {HeaderBufIdxA, FifoAHeadPtr}. Copying the BufIdx instead of relinking avoids a race condition between enqueue and dequeue operations to the same linked list.
- FifoBTailPtr ⇐ FifoAHeadPtr. This allows the old IdxFifo used by the AHeadPtr to be reused as the new BTailPtr.
- FifoAHeadPtr ⇐ FifoANxtIdxFifo.

Thus, enqueue and dequeue operations are entirely RAM-based, requiring only one write and one cycle, unlike conventional implementations, that require at least a read-write-modify of the RAM contents.

As previously mentioned, the invention is embodied as an architecture and a method. While the method has been described incident to the foregoing description of the invented architecture, for clarity, the general steps of the invented method are provided herein below:

- Providing data frames to be processed, where a portion of the frame constitutes a frame header. The provided data frames are received at the ingress port of a network switch;
- Storing each of the frame headers to a RAM buffer;
- Creating a linked list of indexes to said the frame headers, where an index includes a pointer to a buffer occupied by a specific frame, each index constitutes an entry in the linked list, and each entry further includes an index pointer to the next entry in the linked list;
- Pipelining the linked list to a processing unit within said system as an index FIFO, so that the processing unit reads the entries in sequence;
- As entries are read, processing the corresponding data frame; and
- Enqueueing and dequeuing the entries in an index FIFO RAM, so that enqueuing and dequeuing are performed in a single cycle with a single write operation.

Although the invention has been described herein with reference to certain preferred embodiments, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention. Accordingly, the invention should only be limited by the Claims included below.

## CLAIMS

What is claimed is:

1. A method of optimizing data throughput in a data processing system, wherein data to be processed is written to a central buffer and an index FIFO constituting a linked list of indexes to said buffered data is passed to a processing unit within said system, said method comprising the steps of:

providing a plurality of messages to be processed;

writing said data messages to a central buffer;

creating a linked list of indexes to said messages, an index comprising a pointer to a buffer address occupied by a specific message, each index comprising an entry in said linked list, each entry further comprising an index pointer to a next entry in said linked list;

pipelining said linked list to a processing unit within said system as an index FIFO, so that said processor reads entries in sequence;

as entries are read, processing a message indicated by said entry; and

enqueueing and dequeuing said entries in an index FIFO RAM, so that enqueuing and dequeuing are performed in a single cycle with a single write operation.

2. The method of Claim 1, wherein each of said messages comprises a header and wherein said buffer comprises a header buffer, so that said headers are written to said header buffer.

3. The method of Claim 2, wherein said data processing system comprises a plurality of processing units, wherein an index FIFO, said index FIFO comprising said linked list, is processed in series by at least two of said processing units, so that messages corresponding to entries in said index FIFO's are processed accordingly, and wherein said linked list is shared between processing units.

4. The method of Claim 3, wherein said data processing unit comprises a network switch, and wherein said messages constitute data frames.

5. The method of Claim 3, wherein entries in said index FIFO are processed independently of said corresponding messages, so that subsequent entries in said index FIFO may be read while messages corresponding to previous entries are still being processed.

6. The method of Claim 3, wherein pipeline stages are decoupled by allowing entries in said FIFO to be processed independently of each other, so that processing of one entry is unaffected by processing of another.

7. The method of Claim 3, wherein said message processing step comprises the steps of:

reading a message from a location in said header buffer specified by its corresponding entry;

copying said read message into a register; and

using said copy of said message to perform a specified operation.

8. The method of Claim 7, said message processing step further comprising one or both of the steps of:

writing a modified copy of said message to said header buffer location, wherein said read message is modified as a result of said specified operation;

writing a result of said operation to a reserved location in said working RAM.

9. The method of Claim 3, wherein said linked list further comprises a head pointer and a tail pointer, said head pointer designating a first entry in said linked list, said tail pointer designating a last entry in said linked list, said last entry comprising a reserved entry for enqueuing an additional entry.

10. The method of Claim 9, wherein said deqeueing step comprises updating said head pointer to point from a current first entry to a current second entry, so that said current second entry becomes a new first entry.

11. The method of Claim 10, wherein said enqueuing step comprises the steps of:

writing a previously dequeued entry to said reserved entry, wherein an index pointer associated with said previously dequeued entry is reused as a new tail pointer, so that allocation of a new index pointer is unnecessary; and

updating said new tail pointer.

12. An architecture for optimizing data throughput in a data processing system, wherein data to be processed is written to a central buffer, and an index FIFO constituting a linked list of indexes to said buffered data is passed to a processing unit within said system, said architecture comprising:

a plurality of processing units;

a central buffer, wherein data messages to be processed by said processing units are written;

one or more index FIFO's, each of said index FIFO's comprising a linked list of indexes to said messages, an index comprising a pointer to a buffer address occupied by a specific message, each index comprising an entry in said linked list, each entry further comprising an index pointer to a next entry in said linked list;

means for pipelining said linked list to said processing unit within said system as an index FIFO, so that said processor reads entries in sequence, wherein, as entries are read, a message indicated by said entry is processed;

an index FIFO RAM; and

means for enqueuing and dequeuing said entries in said index FIFO RAM, so that enqueuing and dequeuing are performed in a single cycle with a single write operation.

13. The architecture of Claim 12, wherein each of said messages comprises a header and wherein said buffer comprises a header buffer, so that said headers are written to said header buffer.

14. The architecture of Claim 13, wherein said data processing system comprises a plurality of processing units, wherein an index FIFO, said index FIFO comprising said linked list, is processed in series by at least two of said processing units, so that messages corresponding to entries in said index FIFO are processed accordingly, and wherein said linked list is shared between processing units.

15. The architecture of Claim 14, wherein said data processing unit comprises a network switch, and wherein said messages constitute data frames.

16. The architecture of Claim 14, wherein entries in said index FIFO are processed independently of said corresponding messages, so that subsequent entries in said index FIFO may be read while messages corresponding to previous entries are still being processed.

17. The architecture of Claim 14, wherein pipeline stages are decoupled by allowing entries in said FIFO to be processed independently of each other, so that processing of one entry is unaffected by processing of another.

18. The architecture of Claim 14, wherein message are processed by:

reading a message from a location in said header buffer specified by its corresponding entry;

copying said read message into a register; and

using said copy of said message to perform a specified operation.

19. The architecture of Claim 18, wherein said read message is modified as a result of said specified operation, so that said modified message is written to said header buffer location.

20. The architecture of Claim 18, further comprising a working RAM, wherein a result of said operation is written to a reserved location in said working RAM.

21. The architecture of Claim 14, wherein said linked list further comprises a head pointer and a tail pointer, said head pointer designating a first entry in said linked list, said tail pointer designating a last entry in said linked list, said last entry comprising a reserved entry for enqueuing an additional entry.

22. The architecture of Claim 21, wherein said dequeuing means updates said head pointer to point from a current first entry to a current second entry, so that said current second entry becomes a new first entry.

23. The architecture of Claim 21, wherein said enqueuing means writes a previously dequeued entry to said reserved entry, wherein an index pointer associated with said previously dequeued entry is reused as a new tail pointer, so that allocation of a new index pointer is unnecessary, and updates said new tail pointer.

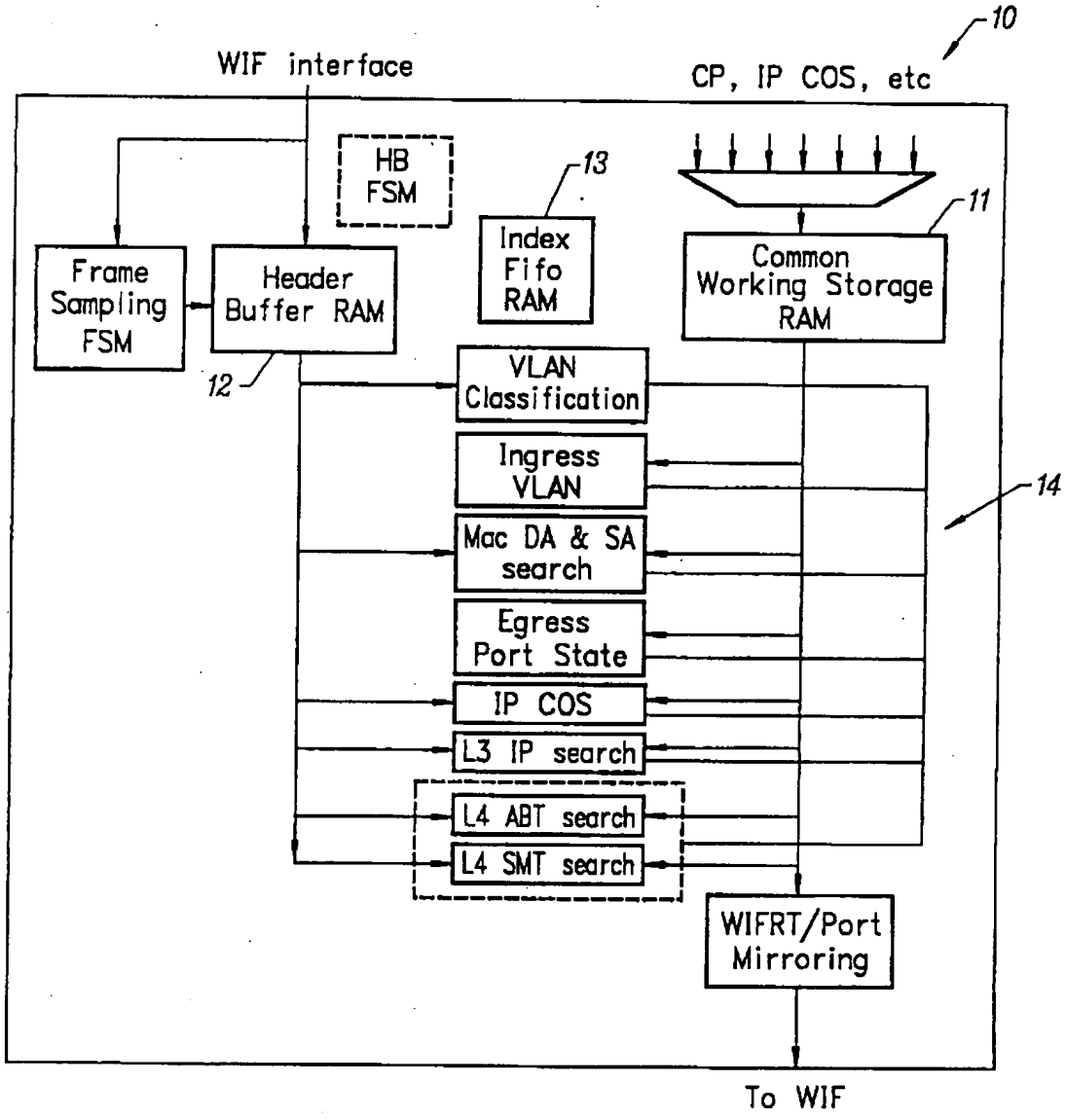
10
WIF interface
CP, IP COS, etc
HB FSM
13
11
Index Fifo RAM
Common Working Storage RAM
Frame Sampling FSM
Header Buffer RAM
12
VLAN Classification
Ingress VLAN
14
Mac DA & SA search
Egress Port State
IP COS
L3 IP search
L4 ABT search
L4 SMT search
WIFRT/Port Mirroring
To WIF

*FIG. 1*

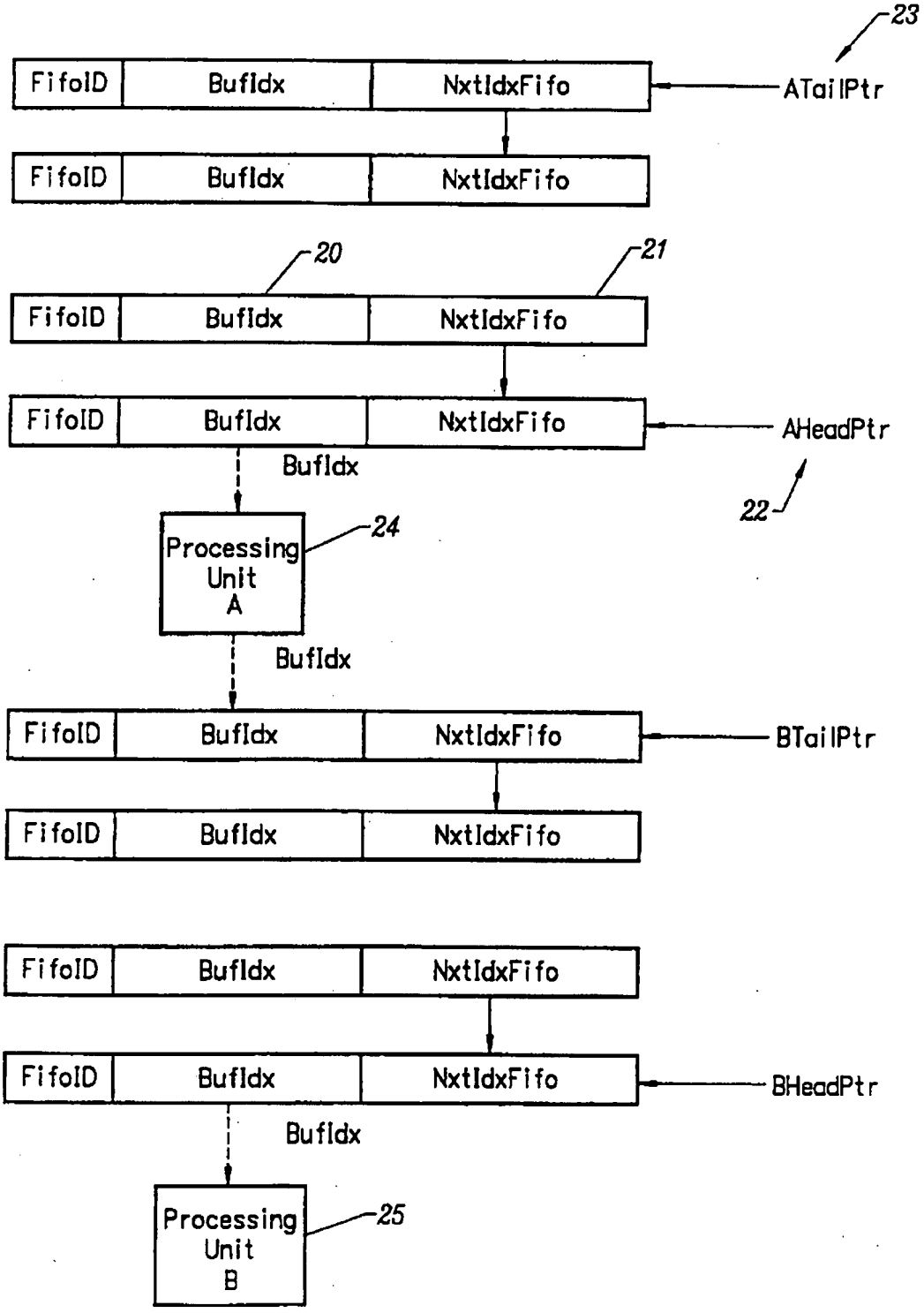
23
FifoID
BufIdx
NxtIdxFifo
ATailPtr
FifoID
BufIdx
NxtIdxFifo
20
21
FifoID
BufIdx
NxtIdxFifo
FifoID
BufIdx
NxtIdxFifo
AHeadPtr
BufIdx
22
Processing Unit A
24
BufIdx
FifoID
BufIdx
NxtIdxFifo
BTailPtr
FifoID
BufIdx
NxtIdxFifo
FifoID
BufIdx
NxtIdxFifo
FifoID
BufIdx
NxtIdxFifo
BHeadPtr
BufIdx
Processing Unit B
25

FIG. 2

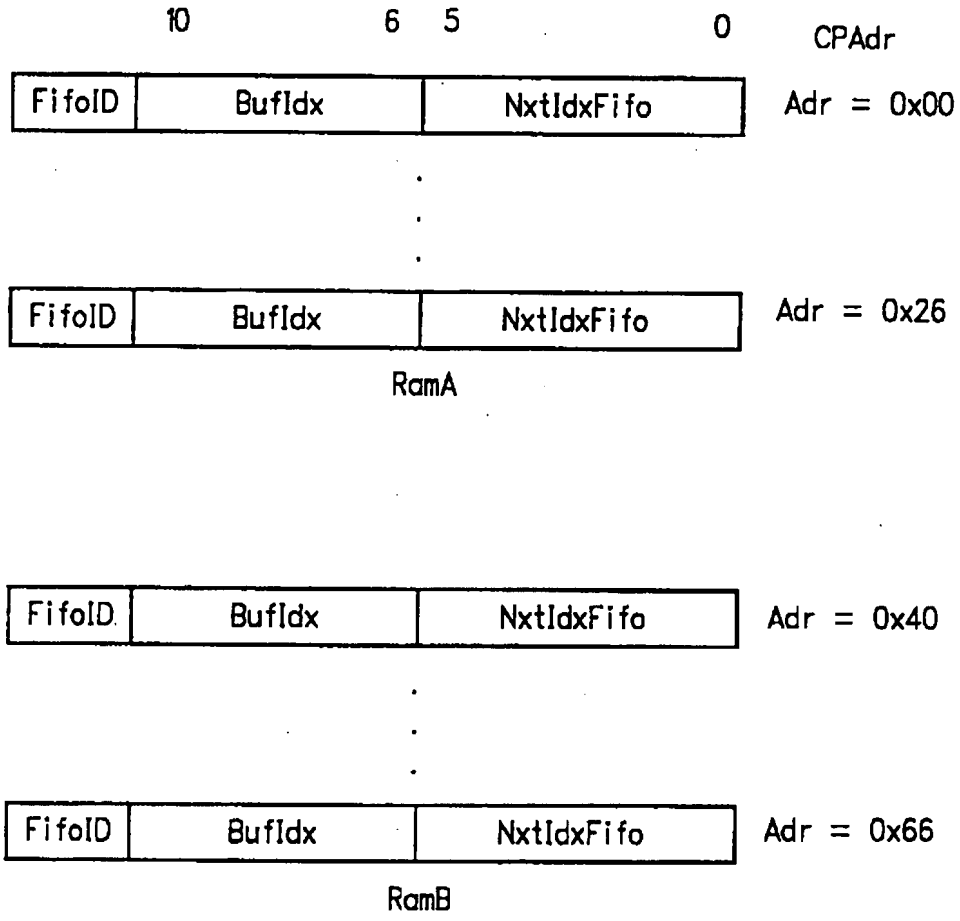
10
6
5
0
CPAdr
FifoID
BufIdx
NxtIdxFifo
Adr = 0x00
FifoID
BufIdx
NxtIdxFifo
Adr = 0x26
RamA
FifoID
BufIdx
NxtIdxFifo
Adr = 0x40
FifoID
BufIdx
NxtIdxFifo
Adr = 0x66
RamB

*FIG. 3*